Computer Science

Development Environment for Teaching a Functional Programming Language: Functional Networked Integrated Environment (FUNNIE)

Brian Howard(*), Computer Science Department, DePauw University, IN 46135 (bhoward@depauw.edu)
Christopher Colvard, Computer Science Department, DePauw University, IN 46135 (ccolvard@depauw.edu)
Matt Ellis, Computer Science Department, Rose-Hulman Inst. of Tech., IN 47803 (ellism@cs.rose-hulman.edu)
Brian McCarty, Computer Science Department, Trinity University, TX 78212 (brian.mccarty@trinity.edu)
Ryan Smith, Computer Science Department, DePauw University, IN 46135 (rksmith@depauw.edu)
Atanas Vlahov, Computer Science Department, DePauw University, IN 46135 (avlahov@depauw.edu)

The long-term aim of the Functional Networked Integrated Environment (FUNNIE) project is to develop a programming environment specifically tuned to the needs of students and instructors, based on a modern high-level functional language such as Haskell. The emphasis is placed on producing tools that are easy to use for students doing coursework rather than being tuned for professional software development. Existing systems like BlueJ, DrScheme, Vital, and Helium try to accomplish this, but are still not very attuned to the needs of beginner students and lack features geared toward class usage and participation. In order for the environment to meet its goal of aiding in the learning process, it should include tools for editing, running, and debugging programs, visualizing data and program structures, managing projects, accessing documentation, as well as allowing networked project collaboration.

The software for FUNNIE is developed in Java on top of the Eclipse Platform, a general purpose integrated development environment from IBM. FUNNIE is being developed as both a plug-in to Eclipse and as a stand alone program for users who do not wish to download the entire Eclipse system just to run FUNNIE. Since FUNNIE is built using the same platform as the Eclipse Java environment, students who are already familiar with Eclipse should have no problem using FUNNIE. It should work on all Java-compatible operating systems running at least the Java 2 Platform, Standard Edition, version 1.4. The networking features use Jabber and CVS, both of which require a dedicated server.

In order to adapt FUNNIE to the needs of beginner students, its language needs to be easy to learn and use. After evaluating several existing systems and their languages, we created our own functional programming language named HasCl, essentially a subset of Haskell with some syntax influenced by Clean and Java. HasCl supports calculations on a broad range of numbers, characters, strings, and booleans, as well as tuple and list types for compound values. The basic unit of a program is the function, which may be defined using recursion and pattern matching. In addition, HasCl supports new types and procedures to be written in Java, allowing instructors or students to extend HasCl in a variety of ways.

FUNNIE plans to support collaboration by allowing students to share code with one another. Shared code will be placed on a public portion of a CVS source-code repository where all user code is stored, and other students will be able to browse shared code and incorporate it into their projects. Additionally, FUNNIE supports chat between multiple users by using the Jabber instant messaging framework.

Future work on FUNNIE will include adding support for collaboration via CVS by building on the existing CVS code in Eclipse, adding support for new types of data (for example, sound files) and extending the HasCl evaluator to optimize evaluation of student code. While there has been much research in the area of optimizing evaluation of functional languages, many techniques would prevent the students from tracing the execution of their code in its original form. Techniques which allow for optimization without changing user code would be especially helpful.

Acknowledgement: This work was supported by National Science Foundation Grant Number EIA-0242293, by the DePauw University Science Research Fellows Program and by an Eclipse Innovation Grant from IBM.